S4H300 – Boost Performance for CDS Views

Randolf Eilenberger, SAP
November 14-16, 2017

EXTERNAL
Speakers

Las Vegas  
September 25 - 29  
Jeff DePianta

Bangalore  
October 25 - 27  
Arun Nair

Barcelona  
November 14 - 16  
Randolf Eilenberger
Disclaimer

The information in this presentation is confidential and proprietary to SAP and may not be disclosed without the permission of SAP. Except for your obligation to protect confidential information, this presentation is not subject to your license agreement or any other service or subscription agreement with SAP. SAP has no obligation to pursue any course of business outlined in this presentation or any related document, or to develop or release any functionality mentioned therein.

This presentation, or any related document and SAP’s strategy and possible future developments, products and or platforms directions and functionality are all subject to change and may be changed by SAP at any time for any reason without notice. The information in this presentation is not a commitment, promise or legal obligation to deliver any material, code or functionality. This presentation is provided without a warranty of any kind, either express or implied, including but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement. This presentation is for informational purposes and may not be incorporated into a contract. SAP assumes no responsibility for errors or omissions in this presentation, except if such damages were caused by SAP’s intentional or gross negligence.

All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, which speak only as of their dates, and they should not be relied upon in making purchasing decisions.
Agenda

Introduction

CDS view complexity

Runtime optimization

Caching

Performance recommendations
Introduction
Target architecture for S/4HANA

S/4HANA Architecture

Fiori Launchpad

- S/4HANA Core App w. Embedded Analytics, Search
- In-App Extension / Industry Verticalization
- SAP / Partner / Customer / Industry Extension App
- Business Objects Cloud (C4A)
- IoT / Big Data App
- LoB Cloud App

S/4HANA Core Cloud & On-premise

- ABAP
  - Transactional Logic
  - Analytics
- HANA Cloud Platform (Cloud), XS Engine Advanced (oP)
  - Transactional Logic
  - Analytics

Virtual Data Model (CDS)

- S/4HANA Core Tables (ECC)
- Extension Tables
- Big Data
- Vora, Messaging

Data Lake

Other Cloud Applications (SAP & 3rd Party)

External Big Data Sources
Why VDM (virtual data model) and CDS (core data services)?

Principle of ONE: model once – use everywhere
- VDM model is leveraged in transactional apps, analytics, APIs, extension apps …
- No need to develop an own model

Execution excellence: run in HANA
- Capitalize on SQL and native HANA features

Understandability: VDM = business semantics
- View and field names, entity relationships, semantic annotations

Visibility and Reuse of Views
- Internal reuse and release to customers and partners

Integration in the robust ABAP development environment
- Use from ABAP SQL, integration with DDIC, development tools, transports
- Authorizations
- Extensibility
- …
SQL processing steps

**select**: *

from T1, T2, T3, T4

where T1.A = T2.A

and T2.B = T3.B

and T3.C = T4.C

and T1.A = 100

---

**HANA SQL Processor**

- **Parsing**
- **Logical plan rewriting**
- **Logical plan enumeration**
- **Physical algorithm enumeration**
- **Cost-based best plan selection**
- **Execution plan**
- **Runtime compilation**

---

**SQL is a declarative language**

Only describes **WHAT** to be processed

**HOW** is determined by SQL processor

---

**SQL Processor is smart, but**

Needs to make decisions in very short time

→ plan might be sub-optimal if too many tables are involved

Cannot fix data model

---

© 2017 SAP SE or an SAP affiliate company. All rights reserved. | EXTERNAL
CDS view complexity
CDS view complexity

CDS view: description of data and its relations. Views can be **stacked and reused**

Danger: stacking can get so complex that it results in performance degradation

Number of execution plan variants raises according to 
$$\# \text{ tables}^5$$

HANA optimizer could fail to find the optimal execution plan
→ high compile and execution times

By using layered CDS views, formerly fast accesses to buffered tables and application caches on ABAP server might be replaced by complex database accesses

Declarative programming style of SQL **can have** a performance drawback since optimizer plans do not take into account the actual attributes of a statement. Cutting away unnecessary execution branches **might** not be done with the same efficiency as with imperative programming in ABAP
Usage of complex CDS views is especially critical in (high volume) transactions and background processing.
Tips & Tricks: how to display complexity of CDS views

Display in ADT:  
→ Open With → Dependency Analyzer

- SQL dependency tree  
- SQL dependency graph  
- Complexity metrics
To cope with complexity: classify CDS views according to their usage

Differentiate which CDS views can be consumed in which situation

- within **business logic** for (high volume) transactions and background processing
- for single object retrieval from UI
- for **analytical reporting**
- in application **code push down** scenarios

Separate transactional data, master data, org data, customizing, and other metadata
This is important to leverage caches on the application server for meta data, customizing and some sort of master data

Provide ‘well tailored’ views instead of the one view for all purposes

Indicate **quality of service** that consumer of CDS view can expect
Classification of CDS views using performance annotations

@ObjectModel.usageType.serviceQuality

quality of service with respect to the expected **performance** of the CDS view

@ObjectModel.usageType.dataClass

ty**pe of data** in CDS view (transactional data, master data, ...)

@ObjectModel.usageType.sizeCategory

set of data which has to be **searched through** in order to compute the result set
CDS views: performance annotations

**serviceQuality**  Each CDS view shall be assigned to one of the following quality categories:
- **A**: the view may be consumed within business logic for high volume transactions or background processing
- **B**: the view may be consumed within business logic for transactions or background processing
- **C**: the view may be consumed from the UI in transactions for single object retrieval
- **D**: the view may be consumed for analytical reporting
- **X**: the view is built to push down application code to HANA

**sizeCategory**  Each CDS view shall have assigned a size category. The size category enables the consumer to judge the possible result set. It reflects the number of rows that has to be searched through to get a result. The labels correspond to the following size categories (expected number of rows in customer production systems):
- **S**: < 1000
- **M**: < 100.000
- **L**: < 10.000.000
- **XL**: < 100.000.000
- **XXL**: > 100.000.000

**dataClass**  To support the decision on cache strategies for higher layers and to enable client side statement routing using these caches, each CDS view shall have assigned a data class. The different data classes correspond to different life time cycles.
- **TRANSACTIONAL**: data is written or changed in high volume transactions
- **MASTER**: data is read, but not written or changed in high volume transactions
- **ORGANIZATIONAL**: data describes the organizational structure of a company and its business processes
- **CUSTOMIZING**: data describes how a concrete business process is executed at the customer
- **META**: data specifies how the system is configured or describes the technical structure of entities
- **MIXED**: data shall be chosen if the CDS-View contains tables with several different of the above types
### serviceQuality of CDS views: requirements and KPIs

<table>
<thead>
<tr>
<th>A</th>
<th>Usage</th>
<th>Expected dataClass</th>
<th>Number of tables</th>
<th>Functions</th>
<th>Aggregation</th>
<th>Data Classes</th>
<th>Buffering</th>
<th>Testing</th>
<th>SQL</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A</strong></td>
<td>may be consumed within business logic for <strong>high volume transactions</strong> or background processing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td><code>SELECT fld FROM view WHERE key LIMIT 1</code></td>
</tr>
<tr>
<td><strong>B</strong></td>
<td>may be consumed within business logic for <strong>transactions</strong> or background processing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td><code>SELECT * FROM view WHERE key LIMIT 1</code></td>
</tr>
<tr>
<td><strong>C</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>D</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>X</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## serviceQuality of CDS views: requirements and KPIs

<table>
<thead>
<tr>
<th>Service Quality</th>
<th>Usage</th>
<th>size Category</th>
<th>Expected dataClass</th>
<th>Number of tables</th>
<th>Functions</th>
<th>Aggregation</th>
<th>Data Classes</th>
<th>Buffering</th>
<th>Testing</th>
<th>Testing</th>
<th>SELECT fld FROM view WHERE key LIMIT 1</th>
<th>SELECT * FROM view WHERE key LIMIT 1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A</strong></td>
<td>may be consumed within business logic for <strong>high volume transactions</strong> or background processing</td>
<td></td>
<td>transactional, master, customizing, org, meta, <strong>but not mixed</strong></td>
<td>≤ 3</td>
<td>no</td>
<td>no</td>
<td>all identical</td>
<td>buffered when all underlying tables buffered</td>
<td>Automatic generic</td>
<td>&lt; 1ms</td>
<td>&lt; 2ms</td>
<td></td>
</tr>
<tr>
<td><strong>B</strong></td>
<td>may be consumed within business logic for <strong>transactions</strong> or background processing</td>
<td>S, M, L, XL or XXL</td>
<td>only if applied to result set (conversions)</td>
<td>≤ 5</td>
<td>no</td>
<td>no</td>
<td>all identical</td>
<td>buffered when all underlying tables buffered</td>
<td>Automatic generic</td>
<td>&lt; 2ms</td>
<td>&lt; 5ms</td>
<td></td>
</tr>
<tr>
<td><strong>C</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>D</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>X</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## serviceQuality of CDS views: requirements and KPIs

<table>
<thead>
<tr>
<th>Service Quality</th>
<th>Usage</th>
<th>size Category</th>
<th>Expected dataClass</th>
<th>Number of tables</th>
<th>Functions</th>
<th>Aggregation</th>
<th>Data Classes</th>
<th>Buffering</th>
<th>Testing</th>
<th>Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>may be consumed within business logic for <strong>high volume transactions</strong> or background processing</td>
<td>S, M, L, XL or XXL</td>
<td>transactional, master, customizing, org, meta, <strong>but not mixed</strong></td>
<td>≤ 3</td>
<td>no</td>
<td>no</td>
<td>all identical</td>
<td>buffered when all underlying tables buffered</td>
<td>Automatic generic</td>
<td>&lt; 1ms</td>
</tr>
<tr>
<td>A</td>
<td>may be consumed within business logic for <strong>transactions</strong> or background processing</td>
<td>S, M, L, XL or XXL</td>
<td>only if applied to result set (conversions)</td>
<td>≤ 5</td>
<td>no</td>
<td>no</td>
<td>all identical</td>
<td>buffered when all underlying tables buffered</td>
<td>Automatic generic</td>
<td>&lt; 2ms</td>
</tr>
<tr>
<td>B</td>
<td>may be consumed from the UI in transactions for <strong>single object retrieval</strong>, may not be used within application logic.</td>
<td>mixed</td>
<td>no aggregation of a huge number of table rows</td>
<td>≤ 15</td>
<td>no</td>
<td>no</td>
<td>all identical</td>
<td>buffered when all underlying tables buffered</td>
<td>Automatic generic</td>
<td>&lt; 10ms</td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Manual</td>
<td>define realistic test case on realistic test data and define KPIs (default &lt; 500ms)</td>
</tr>
<tr>
<td>Service Quality</td>
<td>Usage</td>
<td>size Category</td>
<td>Expected dataClass</td>
<td>Number of tables</td>
<td>Functions</td>
<td>Aggregation</td>
<td>Data Classes</td>
<td>Buffering</td>
<td>Testing</td>
<td>Manual Testing</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------</td>
<td>---------------</td>
<td>-------------------</td>
<td>-----------------</td>
<td>-----------</td>
<td>-------------</td>
<td>--------------</td>
<td>-----------</td>
<td>---------</td>
<td>----------------</td>
</tr>
<tr>
<td>A</td>
<td>may be consumed within business logic for <strong>high volume transactions</strong> or background processing</td>
<td>S, M, L, XL or XXL</td>
<td>transactional, master, customizing, org, meta, but not mixed</td>
<td>≤ 3</td>
<td>no</td>
<td>no</td>
<td>all identical</td>
<td>buffered when all underlying tables buffered</td>
<td>Automatic generic</td>
<td>&lt; 1ms</td>
</tr>
<tr>
<td>B</td>
<td>may be consumed within business logic for <strong>transactions</strong> or background processing</td>
<td>S, M, L, XL or XXL</td>
<td>≤ 5</td>
<td>only if applied to result set (conversions)</td>
<td>no</td>
<td>no</td>
<td>all identical</td>
<td>buffered when all underlying tables buffered</td>
<td>&lt; 2ms</td>
<td>&lt; 5ms</td>
</tr>
<tr>
<td>C</td>
<td>may be consumed from the UI in transactions for <strong>single object retrieval</strong>, may not be used within application logic.</td>
<td>mixed</td>
<td>≤ 15</td>
<td>no</td>
<td>aggregation of a huge number of table rows</td>
<td>no</td>
<td>no</td>
<td>buffered when all underlying tables buffered</td>
<td>Automatic generic</td>
<td>&lt; 10ms</td>
</tr>
<tr>
<td>D</td>
<td>may be consumed for <strong>analytical reporting</strong></td>
<td>XXL</td>
<td>mixed</td>
<td>&lt; 100</td>
<td>no</td>
<td>no</td>
<td>all identical</td>
<td>buffered when all underlying tables buffered</td>
<td>Manual</td>
<td>define realistic test case on realistic test data and define KPIs (default &lt; 500ms)</td>
</tr>
<tr>
<td>X</td>
<td>is built to <strong>push down application code</strong> to HANA</td>
<td>mixed</td>
<td>mixed</td>
<td>mixed</td>
<td>no</td>
<td>no</td>
<td>all identical</td>
<td>buffered when all underlying tables buffered</td>
<td>Manual</td>
<td>As class D or improved performance and throughput compared to implementation without code push-down</td>
</tr>
</tbody>
</table>
CDS view categorization: on premise guide

SAP S/4HANA Requirements for Partner Solutions: On Premise Guide (or here: SAP Help Portal)

VBR-S4H-PERF-01: Categorize CDS Views by Using Performance Annotations and Verify Their Compliance with the Associated Response Time KPIs

There are three categories of characteristics which have to be assigned the annotations `usageType` for the domain `ObjectModel`. These are:

1. `ObjectModel.usageType.serviceQuality`
2. `ObjectModel.usageType.sizeCategory`
3. `ObjectModel.usageType.dataClass`
Runtime Optimization
Complexity: design time vs. execution time

**CDS View**: network of projected, joined, and unioned views and tables

**HANA execution plan**: network of operations like searches, joins, limits

**Tool to analyze**: HANA PlanViz
Tips & Tricks: how to get the HANA execution plan

In ST05 performance trace:

In HANA studio:
Performance expectations: filter on a single table

```sql
select RLDNR, RBUKRS, GJAHR, BELNR, DOCLN
from ACDOCA
where RCLNT = '910' and AWREF = '5000002072' limit 1
```

Filter on high cardinality attribute: **0.3ms on 140m records**

Access supported by secondary index:
Performance expectations: joining large data sets

```sql
select SHKZG, AUGDT, BUDAT, a.BUKRS, count(*)
from BSEG as a
left outer join BKPF as b
on a.MANDT = b.MANDT and
a.GJAHR = b.GJAHR and
a.BUKRS = b.BUKRS and
a.BELNR = b.BEKNR
where a.MANDT = '100' and KOART = 'K' and
  BUDAT >= '20170101' and BUDAT < '20180101'
group by SHKZG, AUGDT, BUDAT, a.BUKRS
```

1.5 seconds for 5.5m joined records

Note: number of joined records << number of table entries
BSEG: 315m, BKPF: 67m entries

→ Filter is pushed down, join is executed after filtering
Important for good performance: filter push down

Reduction of data set on earliest stage of processing … and in all branches of execution

- "Push down" of filters (from CDS view parameters and WHERE-clause)
- … to joined branches through join conditions (ON-clauses)

vs. big intermediate result sets (residing in ‘internal tables’ in HANA memory)

- joins, calculations, aggregation on intermediate results
  scale linear with amount of data

- Rule of thumb: joining 1 million entries in intermediate results takes 1 second
Example: left outer join (LoJoin) - ideal situation

- Filters are restrictive → selected data set is small
- Expensive operations on intermediate results with maximal reduced data set
Example: left outer join (LoJoin) – what can go wrong

- No (selective) filters → selected data set is large
- Expensive operations on huge intermediate result set

- Large intermediate result set
- Exposed non NULL preserving calculated fields
- Join in row store

Filter not pushed to right side:
- Missing field in ON-clause or inefficient push:
- Calculated field in ON-clause
  - “>“, “<“, “OR“ in ON-clause

- No (selective) filters
  - Filter not selective
  - Filter on calculated field

- Filter not at all

Join, calculation, aggregation on huge data set

Table / View L

Table / View R

View: LoJoin

L ⟜ R
Tips & Tricks: create statement for CDS view

SQL representation of a CDS view from ADT (ABAP development tools)
→ can be replaced in SQL statements / other CDS views to test modifications
Example: failed filter pushdown

Example of a predicate derivation issue

Selective filter is not pushed down to huge ACDOCA table

• Q: select ... from ZrevTeLoJoin1
  where REFDOCNR = '1900077097' ...

• Problem:

The filter cannot be pushed to right side
due to a missing join condition:

"STAT"."AWREF" = "ADOC"."BELNR"

• Attention: in case of redundant fields (e.g. posting date on line items) specify these in the ON-clause:

ACDOCA-H_BUDAT = BKPF-H_BUDAT

CREATE VIEW "ZREVTELOJOIN1" AS
SELECT
 "STAT"."MANDT" AS "MANDT",
 "ADOC"."RLDNR" AS "LEDGER",
 "ADOC"."BUKRS" AS "COMPANYCODE",
 "STAT"."GJAHR" AS "FISCALYEAR",
 "STAT"."AWREF" AS "REFDOCNR"
FROM "FAAT_DOC_IT" "STAT"
LEFT OUTER JOIN "ACDOCA" "ADOC" ON
 "STAT"."BUKRS" = "ADOC"."RBUKRS" AND
 "STAT"."ANLN1" = "ADOC"."ANLN1" AND
 "STAT"."ANLN2" = "ADOC"."ANLN2" AND
 "STAT"."GJAHR" = "ADOC"."GJAHR" AND
 "STAT"."MANDT" = "ADOC"."RCLNT" )
Example: exposed non NULL preserving calculated field

```sql
select a.vbeln, b.posnr, objnr_x, stat_x, stsma
from vbak as a
left outer join vbap as b
on b.vbeln = a.vbeln
left outer join (select c.objnr as objnr_x,
                case
                  when c.stat like 'I%' then c.stat
                else ''
                end stat_x,
                d.stsma as stsma
from jest as c
inner join jsto as d
on d.objnr = c.objnr )
on objnr_x = a.objnr
where a.vbeln = '0000000062'
limit 10;
```

```
select a.vbeln, b.posnr, objnr_x, 
    case
    when stat_x like 'I%' then stat_x
    else ''
    end stat_x, stsma
from vbak as a
left outer join vbap as b
on b.vbeln = a.vbeln
left outer join (select c.objnr as objnr_x, c.stat as stat_x, 
                d.stsma as stsma
from jest as c 
inner join jsto as d
on d.objnr = c.objnr )
on objnr_x = a.objnr
where a.vbeln = '0000000062'
limit 10;
```

<table>
<thead>
<tr>
<th>VBELN</th>
<th>POSNR</th>
<th>OBJNR_X</th>
<th>STAT_X</th>
<th>STSMA</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000000062</td>
<td>000010</td>
<td>V8000000006200000001</td>
<td>0002</td>
<td>ZLEGAORD</td>
</tr>
<tr>
<td>0000000062</td>
<td>000010</td>
<td>V8000000006200000002</td>
<td>0002</td>
<td>ZLEGAORD</td>
</tr>
</tbody>
</table>

Statement 'select a.vbeln, ...' successfully executed in **14:45.583 minutes** (14:45.621 minutes)
Fetched 2 row(s) in 0 ms 78 µs (0 ms 0 µs)

```sql
select a.vbeln, b.posnr, objnr_x, 
    case
    when stat_x like 'I%' then stat_x
    else ''
    end stat_x, stsma
from vbak as a
left outer join vbap as b
on b.vbeln = a.vbeln
left outer join (select c.objnr as objnr_x, c.stat as stat_x, 
                d.stsma as stsma
from jest as c 
inner join jsto as d
on d.objnr = c.objnr )
on objnr_x = a.objnr
where a.vbeln = '0000000062'
limit 10;
```

<table>
<thead>
<tr>
<th>VBELN</th>
<th>POSNR</th>
<th>OBJNR_X</th>
<th>STAT_X</th>
<th>STSMA</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000000062</td>
<td>000010</td>
<td>V8000000006200000001</td>
<td>0002</td>
<td>ZLEGAORD</td>
</tr>
<tr>
<td>0000000062</td>
<td>000010</td>
<td>V8000000006200000002</td>
<td>0002</td>
<td>ZLEGAORD</td>
</tr>
</tbody>
</table>

Statement 'select a.vbeln, ...' successfully executed in **34 ms** 332 µs (32 ms 152 µs)
Fetched 2 row(s) in 0 ms 44 µs (0 ms 0 µs)
Limit push down and aggregation push down

Limit push down

A lot of operations break the limit push down

- Order by: first sort, then limit → problem for calculated fields
- Distinct operator, any aggregation
- cardinality changing joins (e.g. LoJoin TO MANY)

Aggregation push down

Rounding (and other functions) stop optimizer from pushing down aggregations

Currency conversion and all arithmetic operations except +/- contain implicit rounding functions
Caching
Classical Golden Rules of database programming

Excellent performance has several aspects

- Fast response times
- Low resource consumption
- Scalability (linear or better)
- Consistent, predictable response time and resource consumption
  - Minimal dependency on data volumes
  - Minimal dependency on data distribution

What is needed for Fiori and HANA

- Still indexes and hints, but different index design
- Static and dynamic cached views, tile cache, extract table
- Data aging, partition pruning and data skipping
- Star schema design and de-normalization (e.g. ACDoca, MATDOC)
- Query offloading

Classical Golden Rules

- Keep the result sets small
- Minimize the amount of transferred data
- Minimize the number of data transfers
- Minimize the search overhead
- Keep load away from the database

What does it mean for HANA?
Caches

Use caches to …

• reduce CPU consumption to save system resources
• increase overall system throughput
• reduce the response time of queries
Use case: value cache for KPI values on Fiori launchpad

- Launchpad with KPI tiles containing highly aggregated results
- In the morning, KPIs would be concurrently calculated for many users
- Do all KPIs need to be up-to-date after every reload of the launchpad?

<table>
<thead>
<tr>
<th>Accounts Receivable - Analytics</th>
<th>Dispute Management - Analytics</th>
<th>Current Settings</th>
<th>Collection Management</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overdue Receivables Today</td>
<td>Reprocessing Rate of Incoming Payments</td>
<td>Total Receivables Today</td>
<td>Future Receivables Today</td>
</tr>
<tr>
<td>100%</td>
<td>C now</td>
<td>10.75 B USD</td>
<td>0.00 USD C now</td>
</tr>
<tr>
<td>C 1 hr. ago</td>
<td></td>
<td>C now</td>
<td>C 2 hr. ago</td>
</tr>
</tbody>
</table>

Display of data age and refresh button
Options in applications: KPI tile caching

An additional (new) ABAP-based service (1) provides caching for KPI tile data (KPI value etc.). It stores its content in an ABAP DDIC table.

KPI tiles (2) first read from the cache and display the cached value. Only if this value is expired (or non-existent) a HANA call starts to update the cache.

The same happens if the user refreshes manually (per individual tile) or navigates to the drill-down app and back.
Caching options in HANA

- **Static Cached Views** (HANA 2.0 SPS 00) = **Result Cache** (HANA 2.0 SPS 01) allow to precompute and reuse the results of complex aggregation queries
  
  **Limitations**: results are only refreshed every k minutes (configurable; minimum of 30 minutes recommended), applications need to deal with outdated results

- **Dynamic Cached Views** = **Dynamic Result Cache** (HANA 2.0 SPS 01) always provide fresh data
  
  **Limitations**: can only be applied to simple queries operating on a single table

- **Active/Active (read only) configuration of HANA** processes on redundant copy of data with additional computing capacity
  
  **Limitations**: copy is not 100% in synch with the source (typical delay of a couple of seconds); does not reduce the resource consumption

This is the current state of planning and may be changed by SAP at any time.
Static cached views / result cache

Static cached views

- query result of a view is cached in HANA memory and refreshed periodically
- configurable data retention period (how ‘old’ data may become)
- cached views are based on **aggregated results** (SUM, MIN, MAX and COUNT supported)
- optimizer uses the aggregation type of each column in the top-most select statement as the basis for the cached result

**Query** must match with the definition of the result cache

- The query must reference the same columns (or a subset of the columns) in the cache
- The query must explicitly reference the view or a sub view referenced by the view
- The predicate in the query must be the same as that used in creating the cached view; any additional conditions which are added must be more restrictive filters
Static cached views / result cache

Examples

Aggregation

This example creates a view and then uses the alter view syntax to add the view to the cache with a retention period of 120 minutes.

```sql
CREATE VIEW SIMPLE_VIEW AS
(SELECT A, SUM(KF1) AS KF1, MIN(KF2) AS KF2, MAX(KF3) AS KF3
 FROM SIMPLE_TABLE GROUP BY A)

ALTER VIEW SIMPLE_VIEW ADD CACHE RETENTION 120 OF A, KF1, KF2, KF3, KF4;
```

In the following example queries the first two statements consistently use the same aggregated values as defined in the query and can exploit the cache. The third example cannot use the cached data because it requests unaggregated details which are not included in the cache:

```sql
SELECT SUM(KF1) FROM SIMPLE_VIEW;

SELECT SUM(KF1), MIN(KF2), MAX(KF3) FROM SIMPLE_VIEW GROUP BY A;

SELECT KF1, KF2, KF3 FROM SIMPLE_VIEW;
```
Dynamic cached views (DCV) / dynamic result cache

- Delta-enabled (=incrementally updated) cache in HANA column store, always up-to-date on read access
- Aggregation view on one column table (planned to extend to 2 tables with inner join)
- released with HANA 2 SPS 00, but not yet for customers
- automatic view matching to allow use of cache in queries based on different views
- additional filter conditions to 'partition' table into smaller chunks to keep to-be-searched-thru data set small
- targeting at particular OLTP processes that aggregate data from very large tables (ACDOCA, MATDOC)
- query plan rewriting in case cache can be used
- not directly related to HANA main and delta memory and merge operation
- has to support multi-version concurrency control and can block garbage collection for MVCC

This is the current state of planning and may be changed by SAP at any time.
Active/Active (read only) configuration

• For S/4HANA on premise
• productive SAP HANA database and an additional ‘hot standby’ SAP HANA database
• **high availability strategy** (business continuity in case of a failure of primary HANA database)
• secondary HANA database retrieves data posted on the primary node by **log replay**
• data delay is typically **one second up to 15 seconds**
• two SAP HANA databases, one of which would be idle for most of the time
• ➞ Analytical UIs addressing SAP Analytical Engine in the S/4HANA ABAP server are allowed to read from secondary SAP HANA database
• SAP Fiori apps based on SAP Smart Business use ‘**cache duration**’ field in tile configuration
• Other SAP Fiori apps use app descriptor
Performance recommendations
Performance recommendations

- Keep CDS views simple (in particular serviceQuality A and B = #BASIC views)
- In transactional processing, only use simple CDS views accessed via CDS key
- Expose only required fields – define associations to reach additional fields when requested
- Perform expensive operations (e.g. calculated fields) after data reduction (filtering, aggregation)
- Avoid joins and filters on calculated fields
- Test performance of CDS views. Test with reasonable (= realistic) test data
- Analyze accesses to more complex views with HANA PlanViz to see whether filters are pushed down to all branches
- Stay tuned on caching possibilities of SAP HANA and Fiori apps
SAP TechEd Online

Continue your SAP TechEd education after the event!

Access replays of
– Keynotes
– Demo Jam
– SAP TechEd live interviews
– Select lecture sessions
– Hands-on sessions
– …

http://sapteched.com/online
Further information

Related SAP TechEd sessions
S4H719 - How to Analyze and Optimize a Slow Query on SAP HANA with a Plan Visualizer
S4H720 - Performance Optimization for CDS Views with Filter Pushdown

SAP Public Web
scn.sap.com
www.sap.com

SAP Education and Certification Opportunities
service.sap.com/performance
service.sap.com/benchmark
service.sap.com/sizing
www.sap.com/education

Watch SAP TechEd Online
www.sapteched.com/online
Thanks for attending this session.

Feedback

Please complete your session evaluation for S4H300.

Contact information:

Randolf Eilenberger
SAP PA&TS Performance and Scalability
randolf.eilenberger@sap.com
© 2017 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company.

The information contained herein may be changed without prior notice. Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

In particular, SAP SE or its affiliated companies have no obligation to pursue any course of business outlined in this document or any related presentation, or to develop or release any functionality mentioned therein. This document, or any related presentation, and SAP SE’s or its affiliated companies’ strategy and possible future developments, products, and/or platform directions and functionality are all subject to change and may be changed by SAP SE or its affiliated companies at any time for any reason without notice. The information in this document is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, and they should not be relied upon in making purchasing decisions.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies. See http://global.sap.com/corporate-en/legal/copyright/index.epx for additional trademark information and notices.